UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
09/702,462	10/31/2000	Alan L. Davis	TI-30558	1089

23494 7590 03/17/2004

TEXAS INSTRUMENTS INCORPORATED
P O BOX 655474, M/S 3999
DALLAS, TX 75265

EXAMINER	
HUISMAN, DAVID J	

ART UNIT	PAPER NUMBER
2183	

DATE MAILED: 03/17/2004

7

Please find below and/or attached an Office communication concerning this application or proceeding.

Office Action Summary	Application No. 09/702,462	Applicant(s) DAVIS ET AL.	
	Examiner David J. Huisman	Art Unit 2183	

-- The MAILING DATE of this communication appears on the cover sheet with th corresp nd nce address --

Peri d for R ply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE 3 MONTH(S) FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If the period for reply specified above is less than thirty (30) days, a reply within the statutory minimum of thirty (30) days will be considered timely.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

Status

1)☒ Responsive to communication(s) filed on *25 February 2004*.

2a)☒ This action is **FINAL**. 2b)☐ This action is non-final.

3)☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

Disposition of Claims

4)☒ Claim(s) *1-21* is/are pending in the application.

4a) Of the above claim(s) _____ is/are withdrawn from consideration.

5)☐ Claim(s) _____ is/are allowed.

6)☒ Claim(s) *1-21* is/are rejected.

7)☐ Claim(s) _____ is/are objected to.

8)☐ Claim(s) _____ are subject to restriction and/or election requirement.

Application Papers

9)☐ The specification is objected to by the Examiner.

10)☒ The drawing(s) filed on *31 October 2000* is/are: a)☐ accepted or b)☒ objected to by the Examiner.

Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).

Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).

11)☐ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

Priority under 35 U.S.C. § 119

12)☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).

a)☐ All b)☐ Some * c)☐ None of:

1.☐ Certified copies of the priority documents have been received.

2.☐ Certified copies of the priority documents have been received in Application No. _____.

3.☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

* See the attached detailed Office action for a list of the certified copies not received.

Attachment(s)

1) ☒ Notice of References Cited (PTO-892)
2) ☐ Notice of Draftsperson's Patent Drawing Review (PTO-948)
3) ☐ Information Disclosure Statement(s) (PTO-1449 or PTO/SB/08) Paper No(s)/Mail Date _____.
4) ☐ Interview Summary (PTO-413) Paper No(s)/Mail Date. _____.
5) ☐ Notice of Informal Patent Application (PTO-152)
6) ☐ Other: _____.

DETAILED ACTION

1. Claims 1-21 have been examined.

Papers Submitted

2. It is hereby acknowledged that the following papers have been received and placed of record in the file: #5. Extension of Time (1 month) as received on 2/25/2004 and #6. Amendment "A" as received on 2/25/2004.

Drawings

3. The drawings are objected to under 37 CFR 1.83(a). The drawings must show every feature of the invention specified in the claims. Applicant has claimed, in claim 21, "determining that a first instruction of a first type of instruction is to be executed in a delay slot of a first branch type instruction, wherein said first instruction calculates a first return address," and "determining that a second instruction of a second type of instruction is to be executed in a delay slot of a first branch type instruction, wherein said second instruction calculates a second return address."

a) From this language, it appears that the first and second instructions are both executed. However, from Fig.6B, the ADDKPC instructions are predicated. That is, only one of the two ADDKPC instructions shown will execute because each instruction's execution is dependent on a different value of B1, but B1 only has one value. Therefore, this idea must be shown or the feature(s) canceled from the claim(s). No new matter should be entered.

b) Looking at Fig.6B, both instructions are of the same type (ADDKPC). Therefore, Fig.6B has not shown a first instruction of a first type and a second instruction of a second type which calculate return addresses. At most, Fig.6B shows a first instruction of a first type and a second instruction of a first type which calculate return addresses.

A proposed drawing correction or corrected drawings are required in reply to the Office action to avoid abandonment of the application. The objection to the drawings will not be held in abeyance.

4. The drawings are objected to because of the following minor informalities: In Fig.6B, LABEL1 630 is associated with instruction 621 (i.e., they're on the same line). Therefore, it appears that if instruction 620 executes, then subroutine 610a will return to instruction 621. However, the drawing shows that subroutine 610a will return to an instruction subsequent to instruction 621. In summation, the examiner believes that LABEL1 630 should be moved down to correspond with the line pointed to by arrow 634. A proposed drawing correction or corrected drawings are required in reply to the Office action to avoid abandonment of the application. The objection to the drawings will not be held in abeyance.

Claim Objections

5. Claims 9 and 18 are objected to because of the following informalities: For increased clarity and claim readability, the examiner recommends replacing "executing a branch to subroutine instruction" with --executing a branch-to-subroutine instruction--. Appropriate correction is required.

Withdrawn Rejections

6. Through amendments, the applicant has overcome the rejections set forth in the office action mailed on October 20, 2003, for claims 1-21. However, upon further consideration, a new ground(s) of rejection is made below.

Claim Rejections - 35 USC § 112

Claim 21 is rejected under 35 U.S.C. 112, second paragraph, as being indefinite for failing to particularly point out and distinctly claim the subject matter which applicant regards as the invention. Applicant has claimed, in claim 21, "determining that a first instruction of a first type of instruction is to be executed in a delay slot of a first branch type instruction, wherein said first instruction calculates a first return address," and "determining that a second instruction of a second type of instruction is to be executed in a delay slot of a first branch type instruction, wherein said second instruction calculates a second return address." It is not clear how two return addresses can be calculated for the same branch without either corruption or one of the return addresses being overwritten. Since both the first and second instructions are executed in the delay slot of a branch, as claimed in claim 21, then two return addresses are generated for the same branch. However, a branch can only have one return address. Further claim 21 claims "arranging an instruction sequence such that the first instruction is executed after the second instruction." From this, the examiner does not see the point of executing the second instruction. When the second instruction executes first, a return address is generated. Then, when the first instruction executes after, another return address is calculated. Does this make the first address void? It appears as though applicant is trying to claim the subject matter shown in Fig.6B.

However, only one return address is calculated due to the predicated execution of the first and second instructions.

Claim Rejections - 35 USC § 102

7. The following is a quotation of the appropriate paragraphs of 35 U.S.C. 102 that form the basis for the rejections under this section made in this Office action:

> A person shall be entitled to a patent unless –
>
> (a) the invention was known or used by others in this country, or patented or described in a printed publication in this or a foreign country, before the invention thereof by the applicant for a patent.
>
> (b) the invention was patented or described in a printed publication in this or a foreign country or in public use or on sale in this country, more than one year prior to the date of application for patent in the United States.

8. Claims 1, 4, 7, and 15 are rejected under 35 U.S.C. 102(a) as being anticipated by Applicant's Admitted Prior Art (as applied in the previous Office Action and herein referred to as AAPA).

9. Referring to claim 1, AAPA has taught a digital processing system comprising a microprocessor (see page 20, lines 8-9), wherein the microprocessor is operable to perform a method for calling a subroutine, the method comprising the steps of:

a) branching to the subroutine by executing a first instruction to provide an address of the subroutine. See Fig.5, instruction 501, and page 20, lines 9-16.

b) calculating a return address by executing a second instruction that is sequentially adjacent to the first instruction to determine a relative return address. This feature is anticipated by AAPA in two different ways:

b1) See Fig.5, instruction 503, and page 20, lines 17-26. When this second instruction completes, a return address will have been formed that is relative to zero. Furthermore,

"Webster's Ninth New Collegiate Dictionary (1990)" has defined the word "adjacent" to mean "not distant" or "nearby". Clearly, instruction 503 is part of an instruction sequence (shown in Fig.5), and instruction 503 is nearby (not distant) to instruction 501. Therefore, instruction 503 is sequentially adjacent to instruction 501.

b2) See Fig.5, instruction 502, and page 20, lines 17-26. When this instruction is executed, a partial return address (lower half) that is relative to zero is formed. Furthermore, instruction 502 is clearly sequentially adjacent to instruction 501. It should be realized that the applicant has not claimed forming a full (complete) return address. Therefore, since instruction 502 forms a partial return address, it still forms a return address as claimed by applicant.

10. Referring to claim 4, AAPA has taught a system as described in claim 1. AAPA has further taught that the second instruction is executed during a delay slot associated with the first instruction. See Fig.5 and page 20, line 27, to page 21, line 3.

11. Referring to claim 7, AAPA has taught a system as described in claim 1. AAPA has further taught during the step of calculating a return address the return address is remotely associated with the second instruction. As described in the rejection of claim 1, the second instruction will form a return address. Therefore, this address is remotely associated with the second instruction because the return address is determined by the second instruction.

12. Referring to claim 15, AAPA has taught a method for calling a subroutine in a digital processing system comprising a microprocessor, the method comprising the steps of:

a) branching to the subroutine by executing a first instruction to provide an address of the subroutine. See Fig.5, instruction 501, and page 20, lines 9-16.

b) calculating a return address by executing a second instruction that is sequentially adjacent to the first instruction to determine a relative return address. This feature is anticipated by AAPA in two different ways:

b1) See Fig.5, instruction 503, and page 20, lines 17-26. When this second instruction completes, a return address will have been formed that is relative to zero. Furthermore, "Webster's Ninth New Collegiate Dictionary (1990)" has defined the word "adjacent" to mean "not distant" or "nearby". Clearly, instruction 503 is part of an instruction sequence (shown in Fig.5), and instruction 503 is nearby (not distant) to instruction 501. Therefore, instruction 503 is sequentially adjacent to instruction 501.

b2) See Fig.5, instruction 502, and page 20, lines 17-26. When this instruction is executed, a partial return address (lower half) that is relative to zero is formed. Furthermore, instruction 502 is clearly sequentially adjacent to instruction 501. It should be realized that the applicant has not claimed forming a full (complete) return address. Therefore, since instruction 502 forms a partial return address, it still forms a return address as claimed by applicant.

13. Claims 1-3, 6-7, and 15-17 are rejected under 35 U.S.C. 102(b) as being anticipated by Iwao, U.S. Patent No. 4,799,151 (as applied in the previous Office Action).

14. Referring to claim 1, Iwao has taught a digital processing system comprising a microprocessor, wherein the microprocessor is operable to perform a method for calling a subroutine, the method comprising the steps of:

a) branching to the subroutine by executing a first instruction to provide an address of the subroutine. See Fig.5, and column 1, lines 15-21, and note that when a BAL (branch and link) instruction is executed, an address to a subroutine is provided.

b) calculating a return address by executing a second instruction that is sequentially adjacent the first instruction to determine a relative return address. See Fig.5, and note that at the end of a subroutine, a return instruction is executed, whereby the return PC is added to a constant provided by the return instruction, so that a target address is formed. For instance, from Fig.5, the initial return address after the BAL instruction is address 521. If the return at the end of block S3 executes, the value 5 is added to the return PC to obtain the address 526 (therefore, the return would cause a return back to block A3 of the program). Furthermore, "Webster's Ninth New Collegiate Dictionary (1990)" has defined the word "adjacent" to mean "not distant" or "nearby". Clearly, the return instruction is part of the instruction sequence shown in Fig.5, and the return instruction is nearby (not distant) to the BAL instruction. Therefore, the return instruction is sequentially adjacent to the BAL instruction.

15. Referring to claim 2, Iwao has taught a system as described in claim 1. Iwao has further taught that the step of calculating a return address comprises the step of adding a relative displacement value provided by the second instruction to a program address value associated with the second instruction. Again, see Fig.5 and note that the return program address is set after the BAL instruction and is associated with a return instruction (in this case, the return program address is 521). The second instruction (return) will optionally add a displacement value to the return program address. This can be seen at the end of block S3 when the return instruction adds 5 to the return program address 521, resulting in a return address of 526.

16. Referring to claim 3, Iwao has taught a system as described in claim 2. Iwao has further taught that the step of calculating a return address further comprises the step of storing the return address in a general-purpose register of the microprocessor. See Fig.3, component 19, and column 4, lines 16-18.

17. Referring to claim 6, Iwao has taught a system as described in claim 1. Iwao has further taught that during the step of calculating a return address, a plurality of second instructions are conditionally executed in response to a predicate value such that the return address is responsive to the predicate value. See Fig.5 and note that a first return or second return instruction will be conditionally executed based on the predicate value of the determination. If the value is true (yes), the return address will be calculated according to the return instruction at the end of block S3. On the other hand, if the value is false (no), the return address will be calculated according to the return instruction at the end of block S2.

18. Referring to claim 7, Iwao has taught a system as described in claim 1. Iwao has further taught during the step of calculating a return address the return address is remotely associated with the second instruction. As described in the rejection of claim 1, the second instruction will form a return address. Therefore, this address is remotely associated with the second instruction because the return address is determined by the second instruction.

19. Referring to claim 15, Iwao has taught a method for calling a subroutine in a digital processing system comprising a microprocessor, the method comprising the steps of:

a) branching to the subroutine by executing a first instruction to provide an address of the subroutine. See Fig.5, and column 1, lines 15-21, and note that when a BAL (branch and link) instruction is executed, an address to a subroutine is provided.

b) calculating a return address by executing a second instruction to determine a relative return address. See Fig.5, and note that at the end of a subroutine, a return instruction is executed, whereby the return PC is added to a constant provided by the return instruction, so that a target address is formed. For instance, from Fig.5, the initial return address after the BAL instruction is address 521. If the return at the end of block S3 executes, the value 5 is added to the return PC to obtain the address 526 (therefore, the return would cause a return back to block A3 of the program).

20. Referring to claim 16, Iwao has taught a method as described in claim 15. Iwao has further taught that the step of calculating a return address comprises the step of adding a relative displacement value provided by the second instruction to a program address value associated with the second instruction. Again, see Fig.5 and note that the return program address is set after the BAL instruction and is associated with a return instruction (in this case, the return program address is 521). The second instruction (return) will optionally add a displacement value to the return program address. This can be seen at the end of block S3 when the return instruction adds 5 to the return program address 521, resulting in a return address of 526.

21. Referring to claim 17, Iwao has taught a method as described in claim 15. Iwao has further taught the step of calculating a return address further comprises the step of storing the return address in a general-purpose register of the microprocessor. See Fig.3, component 19, and column 4, lines 16-18.

Claim Rejections - 35 USC § 103

22. The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

> (a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negatived by the manner in which the invention was made.

23. Claim 5 is rejected under 35 U.S.C. 103(a) as being unpatentable over AAPA, as applied above, in view of Hennessy and Patterson, Computer Architecture - A Quantitative Approach, 2nd Edition, 1996 (as applied in the previous Office Action and herein referred to as Hennessy).

24. Referring to claim 5, AAPA has taught a system as described in claim 1. AAPA has not explicitly taught that the second instruction is executed before the first instruction. However, Hennessy has shown on page 169 that an instruction that is independent of the branch instruction can be executed either before the branch instruction or after the branch instruction (in the delay slot). See Figure 3.28(a). The purpose of executing a second instruction after the first instruction (in a delay slot) is to perform optimization. However, this optimization is not required. Therefore, since a "branch-independent" can be executed either before or after the branch instruction, it would have been obvious to one of ordinary skill in the art at the time of the invention to modify AAPA to execute the second instruction before the first instruction.

25. Claim 8 is rejected under 35 U.S.C. 103(a) as being unpatentable over AAPA, as applied above, in view of Texas Instruments, TMS32010 User's Guide, 1983 (as applied in the previous Office Action and herein referred to as TI).

26. Referring to claim 8, AAPA has taught a system as described in claim 1. AAPA has further taught the step of filling a number of delay slots associated with the first instruction in an instruction execution pipeline of the microprocessor by executing a number of virtual no-operation (NOP) instructions. See Fig.5, instruction 504. AAPA has not taught that the number of virtual NOPs is specified by the second instruction. However, TI has taught the concept of combining multiple instructions into a single instruction. See page 3-7 and note that the LTA instruction combines the LT and APAC instructions. In general, the integration of multiple components is not given patentable weight or would have been an obvious improvement. One of ordinary skill in the art would have recognized that the explicit NOP instruction (504) of AAPA could be combined with the second instruction (503) in order to reduce the amount of explicit instructions that are executed. This would in turn increase execution speeds. As a result, it would have been obvious to one of ordinary skill in the art at the time of the invention to have the second instruction of AAPA specify the amount of virtual NOPs that are to be executed in the branch's delay slots.

27. Claims 9-10, 12, and 18-19 are rejected under 35 U.S.C. 103(a) as being unpatentable over Hochmuth, U.S. Patent No. 5,822,591, in view of Hennessy, as applied above.

28. Referring to claim 9, Hochmuth has taught a digital processing system comprising a microprocessor, wherein the microprocessor is operable to perform a method for forming a relative address, the method comprising the steps of:

a) fetching a sequence of instructions in response to address locations provided by a program counter. It is inherent that instructions are fetched based on an address within the program

counter. Without a program counter, instructions would not be fetched and programs would not be executed.

b) executing a first instruction immediately after executing a branch to subroutine instruction in the sequence of instructions wherein said first instruction forms the relative address. See Fig.1, and note that a branch to subroutine instruction 100 is executed. Upon execution, it will be determined whether branch 100 is taken or not taken. In this particular diagram, if branch 100 is not taken, then conditional branch instruction 104 will immediately follow in execution. It should be realized that applicant has claimed forming a relative address (not a return address and not an return address associated with the branch to subroutine instruction). Therefore, branch 104 will form a relative address, i.e., a jump address.

c) Hochmuth has not explicitly taught executing instruction 104 by using a first address value provided by the program counter as a source operand. However, Hennessy has taught the concept of PC-relative branches. See the first paragraph on page 82. More specifically, Hennessy has taught that the most common way to specify a branch destination is to supply a displacement (or offset) and add it to the program counter (which means that the program counter is a source operand). This is advantageous because the target is often near to the current instruction, and specifying the position relative to the PC requires fewer bits. In addition, such a technique permits code to run independently of where it is loaded. As a result, it would have been obvious to one of ordinary skill in the art at the time of the invention to modify Hochmuth such that instruction 104 is executing by using a first address value provided by the program counter as a source operand, as taught by Hennessy.

29. Referring to claim 10, Hochmuth in view of Hennessy has taught a digital system as described in claim 9. Hennessy has further taught that the step of executing a first instruction comprises the step of combining a displacement value provided by the first instruction with the first address value provided by the program counter to calculate the relative address. Again, see the first paragraph on page 82 of Hennessy.

30. Referring to claim 12, Hochmuth in view of Hennessy has taught a system as described in claim 10.

a) Hochmuth has not explicitly taught an instruction execution pipeline having a plurality of stages, the program counter being associated with a fetch stage of the instruction execution pipeline, and a first functional circuit associated with an execution phase of the instruction execution pipeline. However, Hennessy has shown a basic pipeline structure which includes all of the aforementioned components. See Figure 3.1 on page 130 and Figure 3.4 on page 134. Note that the pipeline is 5 stages where one of the stages is a fetch stage which is associated with a program counter (PC). In addition, the execution stage is associated with a first functional unit (ALU), as is well known in the art. Implementing such a pipeline allows for an increase in instruction level parallelism, which in turn yields higher throughput and faster execution speeds. As a result, it would have been obvious to one of ordinary skill in the art at the time of the invention to implement a pipeline as taught by Hennessy into the system of Hochmuth in order to increase throughput. Also it should be noted that for an instruction to be fetched in a fetch stage, a program counter must be associated with the fetch stage in order to provide a fetch address. In addition, it should also be realized that for in order to perform some operation (which occurs in the execution stage), a functional unit must exist.

b) Furthermore, Hochmuth has not taught FIFO circuitry connected to receive address values from the program counter, the FIFO circuitry operable to delay each address value received from the program counter, wherein the first functional circuit is operable to add the displacement value provided by the first instruction to a first delayed address value provided by the program counter. However Hennessy has taught FIFO circuitry which delays a PC value until it is finally added to a displacement value by a first functional unit. See Figure 3.4 on page 134. It should be noted that the group of pipeline registers (IF/ID, ID/EX, EX/MEM, and MEM/WB) form a FIFO, where data is passed from one register to the next starting with IF/ID. As discussed in the last sentence of the caption of the Figure, the PC information is carried from left to right (in a FIFO fashion) by these registers for branch instructions. The delayed PC of the corresponding branch will arrive at the ALU in the EX stage where it will be added to a displacement value provided by the instruction fetched from the instruction memory. Figure 3.22 of Hennessy on page 163 shows that this same concept can be performed with less delay as well (where the displacement is added one cycle sooner). As described in part (a) above, Hennessy has disclosed that this is circuitry found in a pipelined system and implementing such a pipeline will increase throughput and execution speeds due to the overlapping execution of instructions. As a result, it would have been obvious to one of ordinary skill in the art at the time of the invention to include this FIFO circuitry which delays a PC value.

31. Referring to claim 18, Hochmuth has taught a method for forming a relative address in a digital processing system comprising a microprocessor, the method comprising the steps of:

a) fetching a sequence of instructions in response to address locations provided by a program counter. It is inherent that instructions are fetched based on an address within the program

counter. Without a program counter, instructions would not be fetched and programs would not be executed.

b) executing a first instruction immediately after executing a branch to subroutine instruction in the sequence of instructions wherein said first instruction forms the relative address. See Fig. 1, and note that a branch to subroutine instruction 100 is executed. Upon execution, it will be determined whether branch 100 is taken or not taken. In this particular diagram, if branch 100 is not taken, then conditional branch instruction 104 will immediately follow in execution. It should be realized that applicant has claimed forming a relative address (not a return address and not an return address associated with the branch to subroutine instruction). Therefore, branch 104 will form a relative address, i.e., a jump address.

c) Hochmuth has not explicitly taught executing instruction 104 by using a first address value provided by the program counter as a source operand. However, Hennessy has taught the concept of PC-relative branches. See the first paragraph on page 82. More specifically, Hennessy has taught that the most common way to specify a branch destination is to supply a displacement (or offset) and add it to the program counter (which means that the program counter is a source operand). This is advantageous because the target is often near to the current instruction, and specifying the position relative to the PC requires fewer bits. In addition, such a technique permits code to run independently of where it is loaded. As a result, it would have been obvious to one of ordinary skill in the art at the time of the invention to modify Hochmuth such that instruction 104 is executing by using a first address value provided by the program counter as a source operand, as taught by Hennessy.

32. Referring to claim 19, Hochmuth in view of Hennessy has taught a digital system as described in claim 18. Hennessy has further taught that the step of executing a first instruction comprises the step of combining a displacement value provided by the first instruction with the first address value provided by the program counter. Again, see the first paragraph on page 82 of Hennessy.

33. Claims 11 and 20 are rejected under 35 U.S.C. 103(a) as being unpatentable over Hochmuth in view of Hennessy, as applied above, and further in view of TI, as applied above.

34. Referring to claim 11, Hochmuth in view of Hennessy has taught a system as described in claim 10.

a) Hochmuth has not explicitly taught that the step of executing comprises the step of providing a number of virtual NOP instructions for execution after the step of executing the first instruction. However, Hennessy has taught that branches have delay slots (since the outcome of the branch won't be know for some amount of time). See pages 168-170. Hennessy has further taught that useful instructions are normally scheduled in the branch delay slots. However, if a useful instruction cannot be scheduled, one or more NOPs must be scheduled. See the caption under Fig.3.29 of Hennessy. This technique allows for the reduction or elimination of stalling, as shown in Fig.3.27. As a result it would have been obvious to one of ordinary skill in the art at the time of the invention to modify Hochmuth such that a number of NOPs are provided for execution after the execution of the first instruction, as taught by Hennessy.

b) Furthermore, Hochmuth in view of Hennessy has not taught that the number of virtual NOPs is provided during the execution of the first instruction. However, TI has taught the concept of

combining multiple instructions into a single instruction. See page 3-7 and note that the LTA instruction combines the LT and APAC instructions. In general, the integration of multiple components is not given patentable weight or would have been an obvious improvement. One of ordinary skill in the art would have recognized that the useful delay slot instruction could be combined with the explicit NOP instructions in order to reduce the amount of explicit instructions that are executed. This would in turn increase execution speeds. As a result, it would have been obvious to one of ordinary skill in the art at the time of the invention to have the first instruction of Hochmuth in view of Hennessy specify the amount of virtual NOPs that are to be executed after the branch.

35. Referring to claim 20, Hochmuth in view of Hennessy has taught a system as described in claim 19.

a) Hochmuth has not explicitly taught that the step of executing comprises the step of providing a number of virtual NOP instructions for execution after the step of executing the first instruction. However, Hennessy has taught that branches have delay slots (since the outcome of the branch won't be know for some amount of time). See pages 168-170. Hennessy has further taught that useful instructions are normally scheduled in the branch delay slots. However, if a useful instruction cannot be scheduled, one or more NOPs must be scheduled. See the caption under Fig.3.29 of Hennessy. This technique allows for the reduction or elimination of stalling, as shown in Fig.3.27. As a result it would have been obvious to one of ordinary skill in the art at the time of the invention to modify Hochmuth such that a number of NOPs are provided for execution after the execution of the first instruction, as taught by Hennessy.

b) Furthermore, Hochmuth in view of Hennessy has not taught that the number of virtual NOPs is provided during the execution of the first instruction. However, TI has taught the concept of combining multiple instructions into a single instruction. See page 3-7 and note that the LTA instruction combines the LT and APAC instructions. In general, the integration of multiple components is not given patentable weight or would have been an obvious improvement. One of ordinary skill in the art would have recognized that the useful delay slot instruction could be combined with the explicit NOP instructions in order to reduce the amount of explicit instructions that are executed. This would in turn increase execution speeds. As a result, it would have been obvious to one of ordinary skill in the art at the time of the invention to have the first instruction of Hochmuth in view of Hennessy specify the amount of virtual NOPs that are to be executed after the branch.

36. Claim 13 is rejected under 35 U.S.C. 103(a) as being unpatentable over Hochmuth in view of Hennessy, as applied above, and further in view of Sharangpani et al., U.S. Patent No. 6,237,077 (as applied in the previous Office Action and herein referred to as Sharangpani).

37. Referring to claim 13, Hochmuth in view of Hennessy has taught a system as described in claim 12.

a) Hochmuth in view of Hennessy has not taught a plurality of functional units associated with the execution phase the instruction execution pipeline, wherein the instruction execution pipeline receives a fetch packet containing a plurality of instructions associated with each address value provided by the program counter, and wherein a dispatch stage of the pipeline is operable to provide an execution packet that spans two or more fetch packets. However, Sharangpani has

taught such a system. See Fig.2 and notice the fetch packet which contains a plurality of instructions. And, from Fig.3, it has been taught that the plurality of instructions within the fetch packets are accommodated by a plurality of functional units (M0, M1, I0...BR2). Finally, Sharangpani has taught a dispatch stage (Fig.3) where execution packets are formed. It should be realized from Fig.2 that each fetch packet includes a stop field. The stop field of a first packet indicates whether or not inter-group dependencies exist between the first packet and the next packet. See column 4, lines 4-5. More specifically, if the stop field indicates that no group boundary exists between fetch packets, then instructions from multiple fetch packets are considered to be in the same group (thereby creating a larger execution packet), and instructions in the same group are executed in parallel. With this in mind, it can be seen that the size of an execution packet can vary (it can grow if multiple fetch packets are in the same group (denoted by the stop bit). Dispatch circuitry (Fig.3, components 330 and 340) will check the stop field and for each fetch packet, and route the instructions appropriately. See column 7, lines 59-62. This system and setup, in general, allows multiple instructions to be executed in parallel during the same cycle by multiple functional units, as opposed to executing a single instruction per cycle with a single functional unit. As a result, since more instructions would be executed in parallel, throughput would be higher and it would take less time to execute a program. Consequently, it would have been obvious to one of ordinary skill in the art at the time of the invention to modify Hochmuth in view of Hennessy in view of Sharangpani to speed up execution.

b) Recall from the rejection of claim 12 that Hennessy has taught that the FIFO circuitry will provide a delayed address value associated with a branch instruction. Therefore, it follows that if

the branch instruction is in a fetch packet, then that delayed address value is associated with a fetch packet (this corresponds to the final paragraph of claim 13).

38. Claim 14 is rejected under 35 U.S.C. 103(a) as being unpatentable over Hochmuth in view of Hennessy, as applied above, and further in view of Haataja, U.S. Patent No. 6,137,836 (as applied in the previous Office Action).

39. Referring to claim 14, Hochmuth in view of Hennessy has taught a digital system as described in claim 9. Hochmuth in view of Hennessy has not taught that the digital system is a cellular telephone comprising the components set forth in claim 14. However, Haataja has taught a cellular telephone comprising:

a) an integrated keyboard connected to the CPU via a keyboard adapter. See Fig.8, component 72.

b) a display, connected to the CPU via a display adapter. See Fig.8, component 36.

c) radio frequency (RF) circuitry connected to the CPU. See Fig.8, component 56, and column 7, lines 6-11.

d) an aerial connected to the RF circuitry. See Fig.8, component 54.

It should be realized that Hochmuth in view of Hennessy has taught a system that includes operations that increase the functionality of the system. A person of ordinary skill in the art would have recognized that an improved processor (with more functionality) would lead to the overall improvement of the device in which it is embedded. As shown in Fig.8 of Haataja, and, as is well known in the art, cellular telephones are controlled by some sort of processor. Therefore, it would have been obvious to one of ordinary skill in the art at the time of the

invention to have the incorporate the digital system of Hochmuth in view of Hennessy into a cell phone, as taught by Haataja, in order to improve the overall performance of the cell phone.

Response to Arguments

40. Applicant's arguments filed on February 25, 2004, have been fully considered but they are not persuasive.

41. In the remarks, Applicant argues the novelty/rejection of claims 1 and 15 on pages 19-20 of the remarks, in substance that:

> "AAPA does not describe or suggest calculating a return address by executing a second instruction that is sequentially adjacent to the first branch instruction to determine a relative return address."
>
> "AAPA also does not describe or suggest calculating a return address by executing a second instruction to determine a relative return address."

42. These arguments are not found persuasive for the following reasons:

a) Regarding the first argument, see Fig.5, instruction 503, and page 20, lines 17-26. When this second instruction completes, a return address will have been formed that is relative to zero. Furthermore, "Webster's Ninth New Collegiate Dictionary (1990)" has defined the word "adjacent" to mean "not distant" or "nearby". Clearly, instruction 503 is part of an instruction sequence (shown in Fig.5), and instruction 503 is nearby (not distant) to instruction 501 since it is the second instruction after instruction 501 in the sequence. Therefore, instruction 503 is sequentially adjacent to instruction 501.

b) Regarding the second argument, the second instruction does calculate a relative address. As defined by "The American Heritage® Dictionary of the English Language, 3rd Edition, 1992", to calculate is "to ascertain by computation." And, the same dictionary defines computing as "to

determine by the use of the computer. Therefore, when an instruction is executed on a computer, the computer computes (calculates) a relative return address. In addition, it should be realized that the applicant has not claimed forming a full (complete) return address (complete in the sense of setting all the maximum number of bits of the address. Therefore, see Fig.5, instruction 502, and page 20, lines 17-26. When this instruction is executed, the lower half of the relative return address is set to the value specified by the instruction. Therefore, since instruction 502 forms a partial return address, it still forms a return address as claimed by applicant.

43. In the remarks, Applicant argues the novelty/rejection of claims 1 and 15 on page 20 of the remarks, in substance that:

> "Iwao does not described or suggest calculating a return address by executing a second instruction that is sequentially adjacent to the first branch instruction."

44. These arguments are not found persuasive for the following reasons:

a) "Webster's Ninth New Collegiate Dictionary (1990)" has defined the word "adjacent" to mean "not distant" or "nearby". Therefore, what constitutes as being "adjacent" is not restricted to just the very next instruction and is debatable. For instance, looking at Fig.5 of Iwao, if instruction blocks A1, A2, and A3 comprise 10000 instruction each, and the subroutine is a mere 10 instructions, then one may come to the conclusion that since the return instructions (730 or 750) are approximately 10 instruction separated from the BAL instruction in a program that consists of over 30000 instructions, that the return instructions are nearby (adjacent) to the BAL instruction. The examiner recommends amending the claim from "sequentially adjacent" to something along the lines of where no instructions separate the first and second instructions.

45. Finally, applicant argues on pages 23-25 that the additional references provided by the examiner in the previous Office Action do not remedy the failure of AAPA to describe or suggest calculating a return address by executing a second instruction that is sequentially adjacent to the first branch instruction. However, the examiner asserts that AAPA does not fail in suggesting this for the reasons supplied above.

46. In the remarks, Applicant argues the novelty/rejection of claims 8, 11, and 20 on page 23 of the remarks, in substance that:

> "...the TI reference does not describe or suggest a NOP instruction or any instruction that as part of its functionality specifies a number of virtual no-operation instructions. This applicants respectfully suggest that it would not have been obvious to have the second instruction specify the number of virtual no-operations."

47. These arguments are not found persuasive for the following reasons:

a) The examiner has used TI to show the basic concept of combining multiple instructions. Although a combination including a NOP instruction has not been shown by TI, the concept of combination is clear, and the advantages of combining would be realized by one of ordinary skill in the art. By combining instructions, the amount of explicit instructions that are executed is reduced. This would in turn increase execution speeds. Furthermore, as shown in In re Larson 144 USPQ 347 (CCPA 1965), the integration of multiple components is not given patentable weight or would have been an obvious improvement. Consequently, the examiner feels that such a claim is obvious.

Conclusion

48. Applicant's amendment necessitated the new ground(s) of rejection presented in this Office action. Accordingly, **THIS ACTION IS MADE FINAL**. See MPEP § 706.07(a). Applicant is reminded of the extension of time policy as set forth in 37 CFR 1.136(a).

A shortened statutory period for reply to this final action is set to expire THREE MONTHS from the mailing date of this action. In the event a first reply is filed within TWO MONTHS of the mailing date of this final action and the advisory action is not mailed until after the end of the THREE-MONTH shortened statutory period, then the shortened statutory period will expire on the date the advisory action is mailed, and any extension fee pursuant to 37 CFR 1.136(a) will be calculated from the mailing date of the advisory action. In no event, however, will the statutory period for reply expire later than SIX MONTHS from the date of this final action.

Any inquiry concerning this communication or earlier communications from the examiner should be directed to David J. Huisman whose telephone number is (703) 305-7811. The examiner can normally be reached on Monday-Friday (8:00-4:30).

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Eddie Chan can be reached on (703) 305-9712. The fax phone number for the organization where this application or proceeding is assigned is 703-872-9306.

Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see http://pair-direct.uspto.gov. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free).

DJH
David J. Huisman
March 9, 2004

EDDIE CHAN
SUPERVISORY PATENT EXAMINER
TECHNOLOGY CENTER 2100